Docket No.: 1330.1031

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re the Application of:

Christopher T. VOIGT, et al.

Serial No. 09/366,749 Group Art Unit: 2165

Confirmation No. 3440

Filed: August 4, 1999 Examiner: Samuel G. Rimell

For: A SYSTEM PROVIDING DESKTOP INTEGRATION OF PATENT INFORMATION AND DOCUMENT MANAGEMENT

APPELLANT'S BRIEF

Commissioner for Patents
PO Box 1450
Alexandria, VA 22313-1450

Sir:

In a Notice of Appeal filed December 1, 2004, the applicants appealed the Examiner's June 1, 2004, Office Action finally rejecting claims 25-29 and 34-37.

I. REAL PARTY IN INTEREST

The present application is assigned to American Management Systems, Incorporated (AMS), as evidenced by an Assignment recorded on August 4, 1999, at reel/frame 010160/0367. AMS has recently combined with CGI, with the new company being referred to as CGI-AMS. Therefore, the real party in interest is CGI-AMS.

II. RELATED APPEALS AND INTERFERENCES

Neither the appellant, the appellant's legal representative or the assignee are aware of any other appeals or interferences that will directly affect or be directly affected by, or have a bearing on, the Board's decision in the pending appeal.

III. STATUS OF CLAIMS

Claims 1-24 and 30-33 were canceled during prosecution of the application.

Appealed claims 25-29 and 34-37 have been rejected. These are the only pending claims in the subject application.

An Appendix is attached hereto, and lists each of the pending claims in their current form, along with an indication of the status of the claim.

IV. STATUS OF AMENDMENTS

A Response After Final was filed on November 1, 2004. The claims were not amended in the Response. The Response was entered. There are no unentered amendments filed subsequent to the final rejection.

V. SUMMARY OF INVENTION

The present invention as recited, for example, in independent claim 25, relates to a method of using an Application Programming Interface (API) to enable a healthcare system (HCS) -- executed and used by healthcare workers for viewing healthcare records managed by the HCS -- to access a document management system (DMS) server managing patient documents and making them available for access by plural user terminals, the API providing authorization and access to the DMS server. This method of using an Application Programming interface comprises adding a user interface control to a pre-existing user interface of the HCS by adding to the HCS code for executing functions or methods of the user interface control and the API, thereby creating a modified HCS, where the user interface control is part of or interfaces with the API. See, for example, Figure 12 which shows a user interface 212 with the pre-existing user interface of the HCS, and the disclosure at page 20, lines 21-25.

Furthermore, the invention as recited, for example in independent claim 25, recites interactively logging onto the modified HCS by executing an HCS user authentication process of the HCS which is separate from the API, and based on authentication information related to the HCS user authentication process, automatically authenticating a user to use the DMS server. See, for example, Figure 10 at 256.

Independent claim 25 further recites that after authenticating the user to the DMS server, receiving interactive input by the user interacting with the user interface control to cause the API to submit to the DMS server a request to locate a document managed at the DMS server; and displaying the requested document based on a response from the DMS server and navigating the requested document from within the modified HCS using the user interface control.

Therefore, it is respectfully submitted that independent claim 25 can be understood, for example, from Figure 12 which shows a user interface 212 with the pre-existing user interface of the HCS, and the disclosure at page 20, lines 21-25; Figures 10 at 256; Figures 3 & 12.

VI. ISSUES

Whether claims 25-29 and 34-37 are unpatentable under 35 U.S.C. 103(a) over Judge et al. (U.S. Patent 6,401,138) in view of Official Notice and further in view of Myers et al. (U.S. Patent 5,832,340)?

VII. GROUPING OF CLAIMS

It is respectfully submitted that claims 25-29 and 34-37 do not stand or fall together. Instead, it is respectfully submitted that the claims should be grouped together as follows:

a. Claims 25-28 and 36 stand or fall together.

b. Claims 29 and 34 stand or fall together.

c. Claims 35 and 37 stand or fall together.

VIII. ARGUMENT

As indicated in the "GROUPING OF THE CLAIMS" section of this Appeal Brief, it is respectfully submitted that claims 25-29 and 34-37 do not stand or fall together. Instead, it is respectfully submitted that the claims should be grouped together as follows:

a. Claims 25-28 and 36 stand or fall together. Generally these claims are grouped together because, in addition to various features recited in claims 25-28 and 36, each claim also recites "automatically authenticating a user to use the DMS server" and "interactively logging onto the modified HCS by executing an HCS user authentication process". The importance of these features are discussed in below.

b. Claims 29 and 34 stand or fall together. Generally these claims are grouped together because, in addition to various features recited in claims 29 and 34, each claim also recites "automatically authenticating to the DMS" comprising "the API automatically logging the user to the document managing system".

c. Claims 35 and 37 stand or fall together. Generally these claims are grouped together because, in addition to various features recited in claims 35 and 37, each claim also recites "access and security privileges". The importance of "access and security privileges" is discussed below.

1. Claims 25-28 and 36. As discussed above, claims 25-28 and 36 should be grouped together

The present invention as recited, for example, in claim 25, relates to a method of using an Application Programming Interface (API) to enable a healthcare system (HCS) -- executed and used by healthcare workers for viewing healthcare records managed by the HCS -- to access a document management system (DMS) server managing patient documents and making them available for access by plural user terminals. Also, the API provides authorization and access to the DMS server.

As recited, for example, in claim 25, the method comprises adding a user interface control to a pre-existing user interface of the HCS by adding to the HCS code for executing functions or methods of the user interface control and the API, thereby creating a modified HCS, where the user interface control is part of or interfaces with the API.

As recited, for example, in claim 25, the method comprises interactively logging onto the modified HCS by executing an HCS user authentication process of the HCS which is separate from the API, and based on authentication information related to the HCS user authentication process, automatically authenticating a user to use the DMS server.

Furthermore, as recited, for example, in claim 25, the method comprises that after authenticating the user to the DMS server, receiving interactive input by the user interacting with the user interface control to cause the API to submit to the DMS server a request to locate a document managed at the DMS server.

Moreover, as recited, for example, in claim 25, the method comprises displaying the requested document based on a response from the DMS server and navigating the requested document from within the modified HCS using the user interface control.

Therefore, the present invention is directed to the specific method of using an Application Programming Interface (API) to enable a healthcare system (HCS), and the claims recite specific language directed to such an Application Programming Interface (API) to enable a healthcare system.

A. Judge et al. and Myers, either singly or in combination, fail to disclose, suggest or teach a method of automatically authenticating a user to use the DMS server.

As indicated above, claim 25 recites, amongst other novel features, a method of using an Application Programming Interface (API) to enable a healthcare system comprising, amongst other novel features, **automatically authenticating** a user to use the DMS server. See, for example, Figure 12 which shows a user interface 212 with the pre-existing user interface of the HCS, and the disclosure at page 20, lines 21-25.

Judge fails to disclose, suggest or teach such a method for automatically authenticating a user to use the DMS server. Instead, Judge discloses a facility that supports users switching among different application programs such that these applications retain the same patient context. See Judge at column 1, lines 5-10. In Judge, RegistrationApplication( ) must be done for each application, and the PCI RegisterApplication( ) function registers an application with the PCI. This function must be called prior to an application calling any other PCI function. See Judge at column 3, lines 7-9.

Clearly, the RegistrationApplication( ) in Judge that must be done for each application is different from the method of "automatically authenticating" a user to use the DMS server as recited by Applicant in, for example, claim 25.

In the rejection, the Examiner fails to address this issue of automatically authenticating a user to use the DMS server, as specifically recited in claim 25 of Applicant's invention. However, as indicated above, Judge does not specifically indicate how based on authentication information related to the HCS user authentication process, **automatically authenticating** a user to use the DMS server. Therefore, it is respectfully submitted that Judge fails to disclose or suggest this feature.

Moreover, Myers fails to cure the deficiencies of Judge. More specifically, Myers discloses an invention that relates to an electronic medical record system using a text database to store medical encounter information. A header for each encounter record also uses text to store context information for each encounter record. Each header comprises a plurality of attributes embodied as a field descriptor and a value, bound together as a text object. By binding the field descriptors to the values, each encounter record is complete in itself, without reference to database keys, thereby providing a self-validating record storage system. See Myers at Abstract. The essence of Myers is to provide efficient indexing, storage and retrieval of medical information without degrading the value of the information. See Myers at column 2, lines 35-39. This aspect of Myers is different from Applicant's claimed invention, as recited in, for example, claim 25 that requires automatically authenticating a user to use the DMS server.

Moreover, the Examiner, in the Office Action, failed to respond to or address this feature. Instead, Myers was cited by the Examiner for its alleged teachings of a medical information system that can be distributed into document servers (10, 12, 14) that supply documents to the multiple computer workstations (See Office Action at page 3, paragraph 2).

Therefore, it is respectfully submitted that the rejection is overcome.

Although the above comments are specifically directed to claim 25, it is respectfully submitted that the comments are applicable to rejected dependent claims 26-28 and 36, and thus, claims 26-28 and 36 also patentably distinguish over Judge and Myers.

B. Judge et al. and Myers, either singly or in combination, fail to disclose, suggest or teach a method comprising interactively logging onto the modified HCS by executing an HCS user authentication process.

Claim 25 specifically recites "a method of using an Application Programming Interface (API) to enable a healthcare system (HCS)" comprising, amongst other novel features, "interactively logging onto the modified HCS by executing an HCS user authentication process". Judge fails to disclose such unique novel features.

Instead, Judge discloses a process where different applications must register with the PCI. See Judge at column 3, lines 7-15. This is an application-level registration where the API provides mechanisms by which applications can register their identity and their interest in certain types of data and events. See Judge at Abstract.

In the rejection, the Examiner acknowledges that Judge does not describe the log-in process that permits access to the user interface as shown in FIG. 15 of Judge. However, in the outstanding Office Action, the Examiner asserts that Judge illustrates a log-off function which Examiner contends strongly suggests at the existence of a log-on functionality. Examiner proceeds to take Official Notice contending "that it is well known in the art to have a user interface that includes a log-on request interface."

It is respectfully submitted that even if Examiner's assertions are assumed to be true, still, Judge fails to address the specifics of Applicant's claimed invention, for example, in claim 25, which calls for "interactively logging onto the modified HCS by executing an HCS user authentication process". This is because the log-off function in Judge generally does not relate to "**interactively** logging onto the modified HCS". Neither does the log-off or alleged log-on process disclose "executing an HCS user **authentication process**". Instead, Judge teaches registering applications which is merely permitting applications to learn about other applications that have registered to use the PCI (column 1, lines 45-47), and registering application programs for particular events which is merely "sending notifications to such applications when an update to such stored item occurs". See column 1, lines 47-50.

Moreover, Myers fails to cure the deficiencies of Judge. More specifically, Myers discloses an invention that relates to an electronic medical record system using a text database to store medical encounter information. A header for each encounter record also uses text to store context information for each encounter record. Each header comprises a plurality of attributes embodied as a field descriptor and a value, bound together as a text object. By binding the field descriptors to the values, each encounter record is complete in itself, without reference to database keys, thereby providing a self-validating record storage system. See Myers at Abstract. The essence of Myers is to provide efficient indexing, storage and retrieval of medical information without degrading the value of the information. See Myers at column 2, lines 35-39. This aspect of Myers is different from Applicant's claimed invention, as recited in, for example, claim 25 that requires "**interactively** logging onto the modified HCS" and "executing an HCS user **authentication process**".

Therefore, it is respectfully submitted that the rejection is overcome.

Although the above comments are specifically directed to claim 25, it is respectfully submitted that the comments are applicable to rejected claims 26-28 and 36, and thus, claims 26-28 and 36 patentably distinguish over Judge and Myers.

C. The Examiner's proposed modification of Judge et al. would substantially alter Judge et al. and would render it unfit for its intended purpose.

The Examiner proposes modifying Judge "so as to apply the API and new user interfaces to document server computers (10, 12, 14) instead of just an individual computer and distribute the data to multiple workstations". See Office Action of June 1, 2004 at page 3. Applicant understands this to mean that the patient context information sharing of Judge is proposed to be modified to function as a document server. The Examiner's basis for this modification as asserted in the rejection is that while Judge "does not disclose a server system or multiple computers," Myers teaches that a medical information system can be distributed into document servers that supply documents to multiple computer workstations.

MPEP § 2143.01 (end of section), states that there is no prima facie case of obviousness when the suggested combination of references would require a substantial reconstruction and redesign of the elements shown in the primary reference as well as a change in the basic principle under which the primary reference construction was designed to operate.

MPEP § 2143.01 also states that if a proposed modification would render the prior art being modified unsatisfactory for its intended purpose then there is no suggestion or motivation to make the proposed modification.

The Examiner's proposed modification of Judge would require substantial reconstruction and would render it unfit for its intended purpose. In general, this will be shown by summarizing the relevant claim feature, reviewing the teachings of Judge and its functionality, restating the proposed modification of Judge, expanding on the suggested motive for the modification, and explaining what would be involved in extending the functionality of Judge to a server.

i. Claimed DMS Server

Claim 25, for example, recites a document management server. More specifically, claim 25 recites using an Application Programming Interface (API) to enable a healthcare system (HCS) to access a document management system (DMS) server managing patient documents and making them available for access by plural user terminals. The DMS server is clearly a network-type server in that it receives user requests, manages documents and makes them available to user terminals (clients), authorizes access, etc. Furthermore, document servers are well known in the art as network servers that concurrently serve documents over a network to plural users/terminals.

ii. Teachings and functionalities of Judge

Judge is a system for a single user running different applications on a workstation. Judge allows information to be shared between these applications. If a user on a workstation switches from one application to another, shared memory on the workstation is used to exchange patient context information between the two applications. The patient information is "data about the same patient" (Abstract). For example, shared patient data may be information such as a patient's medical record number, a current time, a department, etc. In other words, if a clinician is using an application to view data about a specific disease of a patient, when the clinician switches to another application, the other application receives, via shared memory, the disease currently active in the first application. The overriding purpose of Judge is to allow a user "to review data about a single patient using more than one program" without having to continually reenter the same data about the patient (column 1, lines 22-24).

The mechanism for inter-application information sharing in Judge is shared memory. As discussed further below, shared memory is available only on a local workstation. Furthermore, shared memory is only a conduit for information. To make the exchange of information between different applications possible, Judge provides an application programming interface (API) which any application may be programmed with. The API abstracts the shared information such as a patient ID, disease, etc. The API also provides an convenient interface to access the shared memory.

The local interprocess or inter-application nature of Judge is also apparent from how it is used. Judge teaches that different healthcare applications may be executing on the same workstation. As is well known in graphical interfaces, it is possible to switch between different applications running on a workstation, where the current application is the focus application. Judge explains in lengthy detail a solution for assuring that switching focus between different applications triggers an exchange of patient context information between those applications. Judge explicitly states that the "PCI embodiment provides a mechanism for changing the system focus from one running application to another running application" (column 17, lines 39-42).

In sum, the function of Judge is to allow a single specific workstation user to automatically share the context of their activity (e.g. working on a particular disease or a particular patient) when switching between different applications on a workstation.

iii. Function of Judge and Suggested Motive for Modification of Judge

The rejection states that the proposed modification "would afford the added functionalities created by the API to be usable across multiple computer systems and a large number of users".

Applicant understands that the suggested motive is that it would have been desirable to make the functionalities of Judge available or usable to multiple concurrent users. In other words, it would have been somehow beneficial for different users on different workstations to know what patient, disease, etc. was the subject/target of another user using an application on another computer.

iv. Modifying Judge to be a Server is Impractical and Would Require Complete Redesign of Judge

Judge discloses a system in which applications on a same workstation share information about the context (particular patient) in which they are being used. So, if a doctor is using a chart application for patient X, and the doctor switches to a pharmacy application, the pharmacy application can automatically orient to patient X; the doctor does not need to tell the pharmacy application to access patient X – this happens automatically.

As discussed above, the mechanism that Judge uses is shared memory. However, shared memory is not practical for implementing a server or for sharing information over a network as a server would. It is well known in the art of computer programming that shared memory is used for communication between processes executing on the same machine under the same operating system instance.

As stated in The Code Project, shared memory can only be used for a client/server systems where the clients and server exist only locally on one machine. ("Since **<u>shared memory may be used only on a single machine</u>**, all processes must run on a same computer. This is an obvious limitation. However, the advantage is a speed of communication (note that speed may be severely compromised by a bad implementation on both sides - client and server ...)", http://www.thecodeproject.com/threads/sharedmemipc.asp, copy included herewith), emphasis added.

Also, as stated at Developer.com, a problem with shared memory is that "[a]ccess to shared memory must be protected from concurrent access which results in data corruption – synchronization". Furthermore, "[shared memory] is a very powerful mechanism to transfer large amount of data between the client and the server. It is also very fast (actual interprocess communication) and secure. However, **<u>it is limited to the local machine - it does not work via a network</u>**" (emphasis added, http://www.developer.com/net/cplus/article.php/632001, copy included).

It is well known in the art that shared memory is not adaptable to a multi-user network server because of the difficulty of providing protection for concurrent access, synchronization, and avoiding corruption of data. In Judge, only one application has focus at one time, and multiple applications do no concurrently access the shared memory at one time; concurrency is not an issue. Rather, the shared memory is accessed in turn when there is a switch to another application. The proposed modification of Judge would require discarding the shared memory design and redesigning it from the ground up. A prima facie case of obviousness has not been made because this kind of substantial redesign and reconstruction has been explicitly held to be non-obvious.

A prima facie case of obviousness has also not been made because the primary mode of operation of Judge would be changed. The primary mode of operation in Judge is single user operation. As discussed above, Judge is designed for the purpose of allowing a single user to share information between two applications that the user is using on a workstation. Its mode of operation is intrinsically single-user. Multi-user document serving is a fundamentally different mode of operation.

Judge cites Hewlett Packard's CareVue product as an exemplary system in which it can be used. However, a review of the CareVue product shows that it is itself a single-vendor client/server system. Clients or workstations (such as those in Judge) use multiple applications to access backend database servers, preferably using ODBC. In other words, the applications in Judge are clients that access database servers (e.g. CareVue database servers). It would not make sense to modify an inter-application helper interface (Judge) to operate as a server, particularly when database servers already handle data sharing between computers/users.

v. Proposed Modification Would Render Judge Unfit For Its Intended Purpose

The purpose of Judge is to allow a user to transfer information between applications *when switching focus from one application to another* (possibly by launching a new application, or interactively selecting another application). Shared memory is used because the functionality need only be local and fast. Modifying Judge to be a server would require an access to a server every time a user switched applications. Even short delays for accessing a server would significantly impair the process of locally changing between applications. Judge et al. chose shared memory because their intended purpose was purely local. Modifying Judge to operate as a server would render it unsatisfactory for its intended purpose of sharing information when local applications change focus. Therefore, there is no suggestion or motivation to make the proposed modification.

In sum, Judge is for a user to automatically share information between the applications that they are locally using. The rejection proposes that making this available for a document server "would afford the added functionalities created by [Judge's shared memory] API to be usable across multiple computer systems and a large number of users". However Judge's functionality – inter-application communication for a single local user – is impracticable for a server and irrelevant to multiple users. There is no reason for one user to share with another user information about which patient they are currently accessing, which disease they are currently researching, etc. One user does not care which patient another user is working on. Substantive data sharing is handled by a backend database server, as is the case with most prior art healthcare systems.

2. Claims 29 and 34 should be grouped together

A. No portion of Judge et al or Myers, either singly or in combination, discloses, suggests or teaches the API automatically logging the user to the document managing system using the log-on information from the user interface.

Claims 29 and 34 specifically recite "the interactive logging onto the modified HCS comprises a healthcare worker providing log-on information to the modified HCS using the user interface of the HCS, and wherein the automatically authenticating to the DMS comprises the API automatically logging the user to the document management system using the log-on information from the user interface". See, for example, Figure 10 at 256. Judge fails to disclose such unique novel features. Instead, Judge discloses a process where different applications must register with the PCI. See Judge at column 3, lines 7-15. This is an application-level registration where the API provides mechanisms by which applications can register their identity and their interest in certain types of data and events. See Judge at Abstract. Moreover, Myers fails to cure the deficiencies of Hama, and merely discloses an invention that relates to an electronic medical record system using a text database to store medical encounter information wherein a header for each encounter record also uses text to store context information for each encounter record.

The rejection notes that "the API performs a logging operation where applications are registered into the PCI. This is also an authentication process since the PCI can issue responses that applications being loaded are successful or invalid". Registering an application with a local service (the API) does not require user authentication. In fact, the purpose of Judge is to allow sharing of Patient Context Information, not authentication information. Columns 17 and 18 in Judge discuss the Switch-to button as merely switching control to the other application. There is no discussion or suggestion of user authentication when switching to those applications.

Furthermore, as shown above, Judge cannot be modified into a server. Therefore, there is not a prima facie case of automatic authentication to a DMS server. The Official Notice only establishes that login authentication is known art. However, automatic authentication based on application switching could present serious problems in the privacy-sensitive healthcare context. Withdrawal of the rejection is further respectfully requested.

Moreover, dependent claims 29 and 34 depend on independent claim 25. As indicated in the arguments above, Judge et al and Myers, either singly or in combination, fail to overcome Applicant's claimed invention as recited in, for example, claim 25. As such, for at least these reasons, it is respectfully submitted that claims 29 and 34 should be allowable.

3. Claims 35 and 37 should be grouped together

A. No portion of Judge et al or Myers, either singly or in combination, discloses, suggests or teaches a method wherein accessing, displaying, and navigating the requested document is based on access and security privileges of a user obtained by the authenticating with the HCS or the DMS server.

Claim 35 specifically recites, amongst other novel features, a method wherein accessing, displaying, and navigating the requested document is based on access and security privileges of a user obtained by the authenticating with the HCS or the DMS server. Judge and Myers do not disclose these features.

Instead, Judge discloses how the PCI_Retrieve( ) function retrieves data from the PCI by copying the data to an application's buffer. The same data may be retrieved by any number of the applications that have registered for that particular type of data. The PCI does not itself interpret the content of the data; it is the applications sharing the data that give particular meaning to the data. It is the responsibility of the applications sharing data through the PCI to be sure to use the same dataId and sz for each type of data being shared. See Judge at column 7, lines 7-15, figure 15. Clearly, Judge fails to disclose how displaying, and navigating the requested document is based on access and security privileges of a user obtained by the authenticating with the HCS or the DMS server. Myers fails to cure any of the deficiencies of Judge.

Moreover, in the rejection, the Examiner asserts that "once a user has logged in, the user has obtain[ed] some degree of security privileges with the system, and access the user interface of FIG. 15 to retrieve documents. It is respectfully submitted that Examiner has failed to provide the basis for such assertions. Neither Judge nor Myers disclose or suggest such features. Also, there are no basis for a person of ordinary skill in the art to be motivated by Judge and/or Myers to disclose a method wherein accessing, displaying, and navigating the requested document is based on access and security privileges of a user obtained by the authenticating with the HCS or the DMS server. And Examiner has failed to provide any reason(s) to the contrary for such motivations.

Therefore, it is respectfully submitted that the fundamental nature of the present invention as recited, for example in claim 35, patentably distinguishes over Judge et al. and Myers.

Although the above comments are specifically directed to claim 35, it is respectfully submitted that the comments are applicable to rejected claim 37, and thus, claim 37 patentably distinguishes over Judge and Myers.

IX. CONCLUSION

In view of the above, it is respectfully submitted that the application is in condition for allowance, and a Notice of Allowance is earnestly solicited.

If any further fees are required in connection with the filing of this Appeal brief, please charge such fees to our Deposit Account No. 19-3935.

Respectfully submitted,

STAAS & HALSEY LLP

Date: May 2, 2005

By: [illegible]
Paul I. Kravetz
Registration No. 35,230

1201 New York Ave, N.W., Suite 700
Washington, D.C. 20005
Telephone: (202) 434-1500
Facsimile: (202) 434-1501

APPENDIX A

1-24. (CANCELLED)

25. (PREVIOUSLY PRESENTED) A method of using an Application Programming Interface (API) to enable a healthcare system (HCS) - executed and used by healthcare workers for viewing healthcare records managed by the HCS - to access a document management system (DMS) server managing patient documents and making them available for access by plural user terminals, the API providing authorization and access to the DMS server, the method comprising:

adding a user interface control to a pre-existing user interface of the HCS by adding to the HCS code for executing functions or methods of the user interface control and the API, thereby creating a modified HCS, where the user interface control is part of or interfaces with the API;

interactively logging onto the modified HCS by executing an HCS user authentication process of the HCS which is separate from the API;

based on authentication information related to the HCS user authentication process, automatically authenticating a user to use the DMS server;

after authenticating the user to the DMS server, receiving interactive input by the user interacting with the user interface control to cause the API to submit to the DMS server a request to locate a document managed at the DMS server; and

displaying the requested document based on a response from the DMS server and navigating the requested document from within the modified HCS using the user interface control.

26. (PREVIOUSLY PRESENTED) A method according to claim 25, wherein the interacting causes the user interface control to access a control object of the API that performs the submitting.

27. (PREVIOUSLY PRESENTED) A method according to claim 26, wherein the control object is a query control object.

28. (PREVIOUSLY PRESENTED) A method according to claim 25, further comprising adding a user interface element to the HCS by adding code to the HCS, where the user interface element performs the displaying and navigating of the requested document.

29. (PREVIOUSLY PRESENTED) A method as recited in claim 25, wherein the interactive logging onto the modified HCS comprises a healthcare worker providing log-on information to the modified HCS using the user interface of the HCS, and wherein the automatically authenticating to the DMS comprises the API automatically logging the user to the document management system using the log-on information from the user interface.

30-33. (CANCELED)

34. (PREVIOUSLY PRESENTED) A method as recited in claim 25, wherein the interactive logging onto the modified HCS comprises a healthcare worker providing log-on information to the modified HCS using the user interface of the HCS, and wherein the automatically authenticating to the DMS comprises the API automatically logging the user onto the document management system using the log-on information from the user interface.

35. (PREVIOUSLY PRESENTED) A method as recited in claim 25, wherein accessing, displaying, and navigating the requested document is based on access and security privileges of a user obtained by the authenticating with the HCS or the DMS server.

36. (PREVIOUSLY PRESENTED) A method as recited in claim 25 further comprising performing chart deficiency completion from the user interface.

37. (PREVIOUSLY PRESENTED) A method as recited in claim 25, further comprising displaying and editing of fields with the requested document based on access and security privileges of a user obtained by the authenticating with the HCS or the DMS server.